

REMARKS

Claims 1 – 3 are pending in the application.

Claims 1 and 3 have been rejected under 35 USC §103(a) as being allegedly unpatentable over “Intrusion Detection Using Static Analysis” (hereinafter “Wagner”) in view of “Static Analysis” (hereinafter “Webb”).

Prior to discussing the rejection, a brief summary of the invention is provided. The present invention is directed to a method of detecting malicious scripts using code with high accuracy through precise static analysis. In accordance with the method, a step of checking whether a series of methods constructing a malicious code pattern exists is performed, followed by a step of determining whether parameters and return values associated between the methods match each other. In stark contrast, conventional static heuristic analysis, only the step of checking whether a series of methods constructing a malicious code pattern exists is performed. That is, conventional static heuristic analysis fails to perform the step of determining whether parameters and return values associated between the methods match each other. This situation leads to false positives since respective methods for use in the self-replication behavior can be frequently used in the general scripts.

In making the rejection, the Examiner contends that, regarding claim 1, Wagner discloses a method for detecting malicious scripts using a static analysis, comprising the step of: checking whether a series of methods constructing a malicious code pattern exist and whether parameters and return values associated between the methods match each other as recited in Claim 1 of the present application. In support of the Examiner position, the Examiner

directs the Applicants attention to page 158, first paragraph of Wagner. The Examiner cites Webb for curing a deficiency in Wagner. Specifically, the Examiner cites Webb for teaching extracting parameters in the rule variables. The Examiner further asserts that the ability to statically analyze “local variables, data structures, and all other data flow” in a script so as to determine if the script is non-hazardous has been long since known in the art, and has even been realized in pre-existing products (the MALPAS system and in particular the “Control Flow Analyzer”, “Data Use Analyzer” and “Information Flow Analyzer” sections. However, for the reasons discussed below the Applicants respectfully disagree and traverse the rejection.

Wagner describes an intrusion detection system that defines a specification of expected application behavior, and a monitoring of actual behavior to determine if the actual behavior deviates from the specification defining the expected behavior. To reduce the potentially huge volume of trace data, Wagner only considers the security-relevant behavior of the application of interest. Accordingly, Wagner only considers system calls that interact with the operating system. Wagner proposes a sequence of models used to specify expected application behavior, i.e., a trivial model, a callgraph model, an abstract stack model, the context-free model and a low-overhead digraph model. In the Office Action, the Examiner makes particular reference to the “abstract stack model” and in particular, the “context-free” model for allegedly teaching an element of claim 1, namely:

wherein the checking step comprises the steps of:

classifying, by modeling a malicious behavior in such a manner that it includes a combination of unit behaviors each of which is composed of sub-unit behaviors or one or more method calls, each unit behavior and method call sentence into a matching

rule for defining sentence types to be detected in script codes and a relation rule for defining a relation between rule variables used in the sentences satisfying the matching rule;

Wagner discloses that the abstract stack model is a refinement of the callgraph model that is in turn a refinement of the trivial model. In the callgraph model, the *ordering* of all possible system calls is taken into account in the modeling process. For ease of model generation, Wagner uses an equivalent representation of the model as a non-deterministic finite automation (NFA). Wagner further discloses that the model is a simple application of control-flow analysis. In this regard, Wagner builds a control flow graph associated with the program source code. Each node of the graph is assumed to execute at most one system call. The abstract model merely refines this process by excluding impossible paths.

In view of the foregoing, it is respectfully submitted that while Wagner admittedly checks whether a series of methods constructing a malicious code pattern exist, the method in which the checking step is performed is **different from** the checking step recited in Claim 1. It is further submitted that, in contrast to the Examiner's assertion, Wagner **does not determine** whether parameters and return values associated between the methods match each other as recited in Claim 1 of the present application.

Claim 1 is notably distinguishable from Wagner in that it particularizes details of a novel-checking step. Wagner, by contrast, merely teaches whether a series of methods constructing a malicious code pattern exists and does not teach or disclose any particularization in this regard. That is, Wagner **does not teach or disclose** the particular steps of combining unit behaviors and method

call sentences into a matching rule for defining sentence types to be detected in script codes and a relation rule for defining a relation between rule variables used in the sentences satisfying the matching rule, as recited in Claim 1. As stated *supra*, conventional static heuristic analysis, only performs the step of checking whether a series of methods constructing a malicious code pattern exists is performed. Wagner is merely one manifestation of such conventional static heuristic analysis.

In the office action, Webb is cited for curing a deficiency in Wagner. More particularly, Webb is cited for disclosing extracting parameters of functions used in the searched code patterns, and storing the extracted parameters in the rule variables. Applicants respectfully disagree. There is no teaching in Webb of storing extracted parameters in the rule variables. In accordance with the method of the invention, as recited in Claim 1, parameters and return values used in the respective methods can be replaced by “rule variables” so that these rule variables can be used in different rules. Therefore, to search for malicious behavior, relations between the rule variables are analyzed in those sentences that satisfy the matching rule. For example, referring to Fig. 6, there is shown a matching rule and a relation rule, whereby parameters and return values are replaced by rule variables, i.e., { ML1, \$1, ML1.\$2 }. By using rule variables instead of parameters and return values, the rule variables may be used in different rules. In essence, it makes the rule matching more generic.

Independent Claim 1 has been amended herein to better define Applicant's invention over the cited references, alone and in combination. Claim

1 now recites limitations and/or features which are not disclosed by these references.

Claim 1 as amended reads:

1. A method for detecting malicious scripts using a static analysis, comprising the step of:

checking whether a series of methods constructing a malicious code pattern exist and whether parameters and return values associated between the methods match each other,

wherein the checking step comprises the steps of:

classifying, by modeling a malicious behavior ~~in such a manner that it includes a~~
to include combination of unit behaviors each of which is composed of sub-unit behaviors or one or more method calls,

converting each identified unit behavior and method call sentence into a matching rule for defining sentence types to be detected in script codes and

generating at least one a relation rule for defining a relation between rule variables used in the sentences satisfying the matching rule;

identifying ~~generating~~ instances of the matching rule by searching for code patterns matched with the matching rule from a relevant script code to be detected, extracting parameters of functions used in the searched code patterns and storing the extracted parameters in the rule variables; and

identifying ~~generating~~ instances of the relation rule by searching for instances satisfying the relations rule from a set of the generated instances of the matching rule.

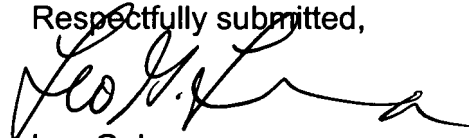
Accordingly, it is believed that Applicant's Claim 1, as amended, is neither taught nor suggested by Wagner in view of Webb, and therefore, withdrawal of the rejections with respect to Claim 1 and allowance thereof is respectfully requested.

Claims 2 and 3 depend from Claim 1, and therefore include the limitations of Claim 1. Hence, for the same reasons given above for Claim 1, Claims 2 and 3 are believed to contain patentable subject matter. Accordingly, withdrawal of the rejection with respect to Claims 2 and 3 and allowance thereof are respectfully requested.

Accordingly, in view of the forgoing remarks, it is respectfully submitted all claims pending herein are in condition for allowance. Please contact the undersigned attorney should there be any questions. A petition for an automatic one-month extension of time for response under 37 C.F.R. §1.136(a) is enclosed in triplicate, together with the requisite petition fee and fee for the additional claims introduced herein.

Early favorable action is earnestly solicited.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Leo G. Lenna", written over the typed name.

Leo G. Lenna
Registration No.: 42,796
Attorney for Applicant(s)

DILWORTH & BARRESE, LLP
333 Earle Ovington Boulevard
Uniondale, New York 11553
Tel. No. (516) 228-8484
Fax No. (516) 228-8516